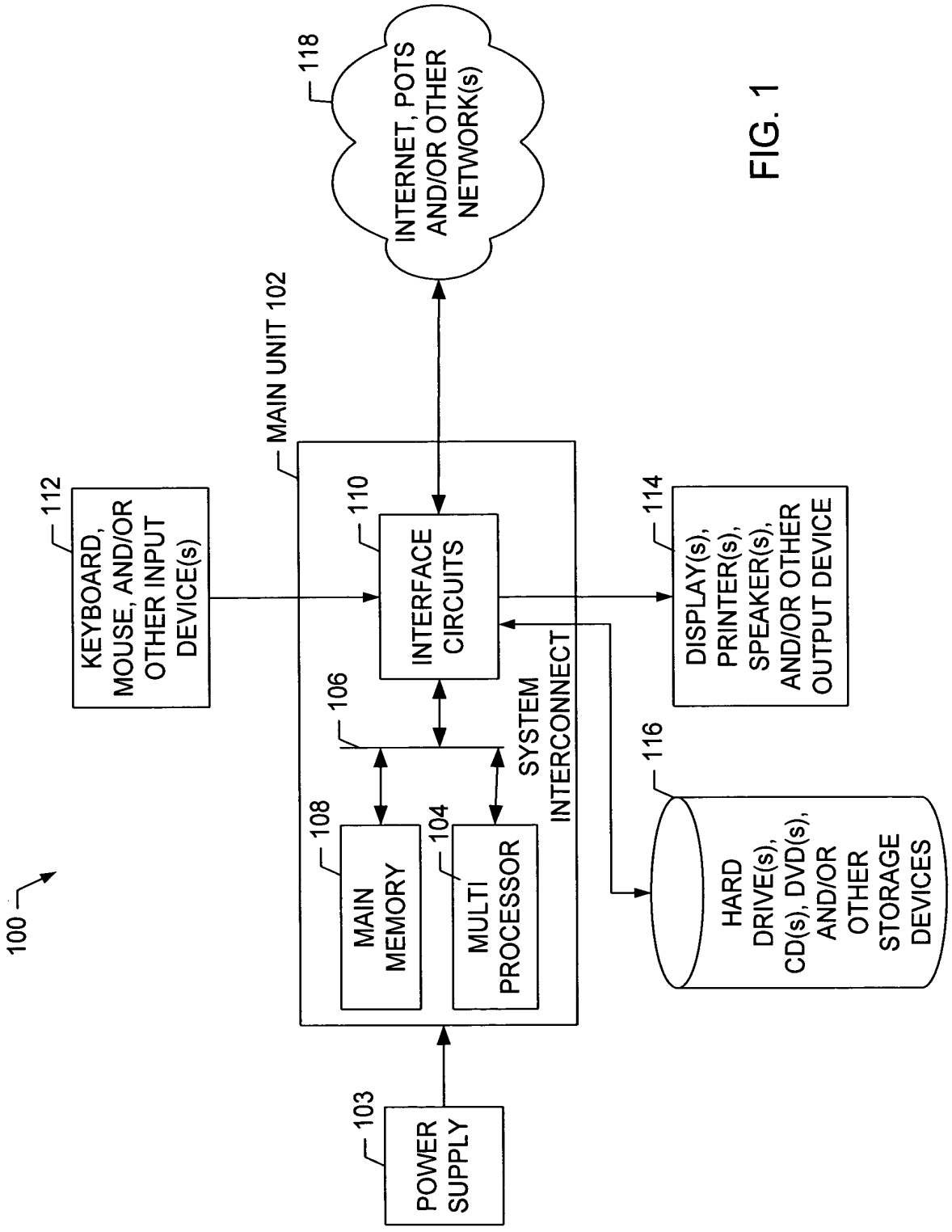

100
MAIN UNIT 102
INTERNET, POTS AND/OR OTHER NETWORK(s)
118
KEYBOARD, MOUSE, AND/OR OTHER INPUT DEVICE(s)
112
INTERFACE CIRCUITS
110
DISPLAY(s), PRINTER(s), SPEAKER(s), AND/OR OTHER OUTPUT DEVICE
114
SYSTEM INTERCONNECT
106
MAIN MEMORY
108
MULTI PROCESSOR
104
HARD DRIVE(s), CD(s), DVD(s), AND/OR OTHER STORAGE DEVICES
116
POWER SUPPLY
103

FIG. 1

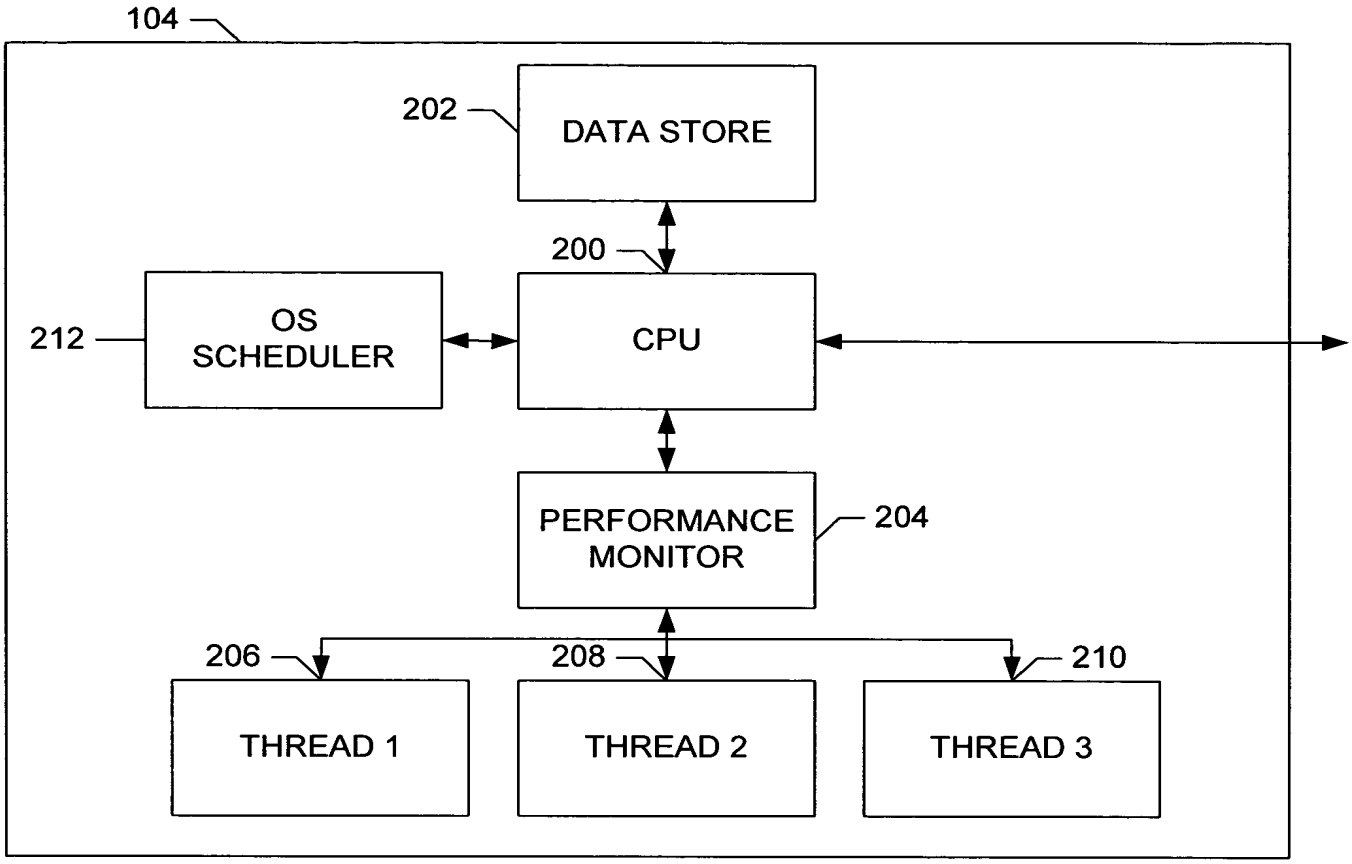

104
202
DATA STORE
200
212
OS SCHEDULER
CPU
PERFORMANCE MONITOR
204
206
208
210
THREAD 1
THREAD 2
THREAD 3

FIG. 2

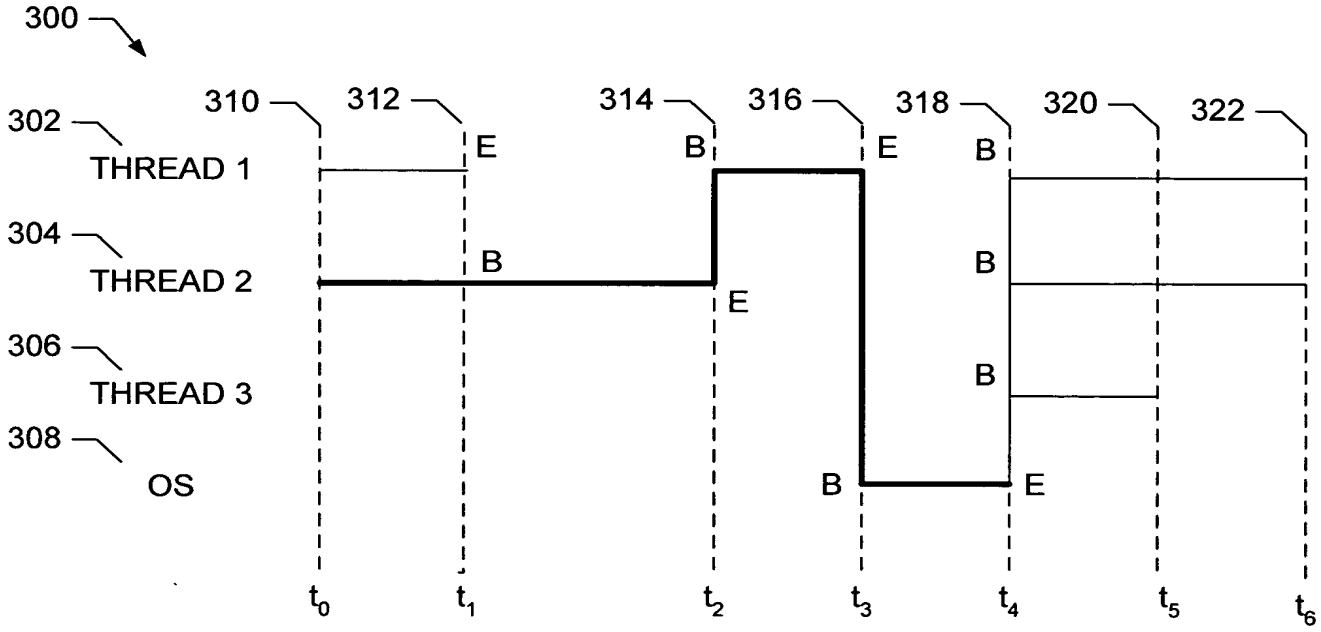

300
310
312
314
316
318
320
322
302
THREAD 1
304
THREAD 2
306
THREAD 3
308
OS
E
B
E
B
E
B
B
E
B
B
B
E
t_0
t_1
t_2
t_3
t_4
t_5
t_6

FIG. 3

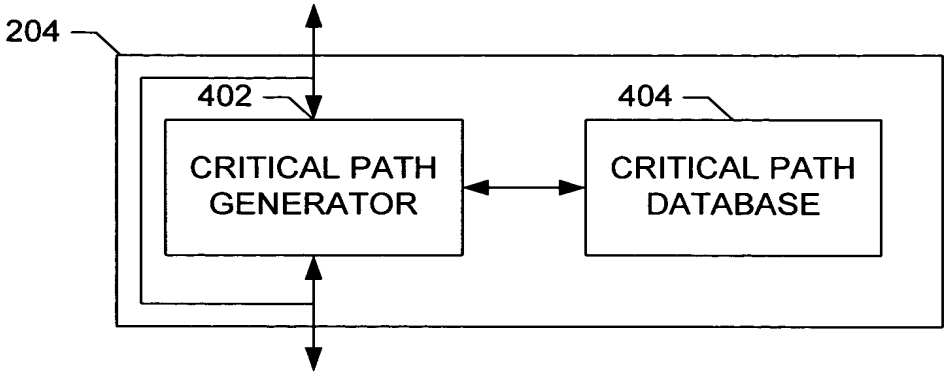

204
402
404
CRITICAL PATH GENERATOR
CRITICAL PATH DATABASE

FIG. 4

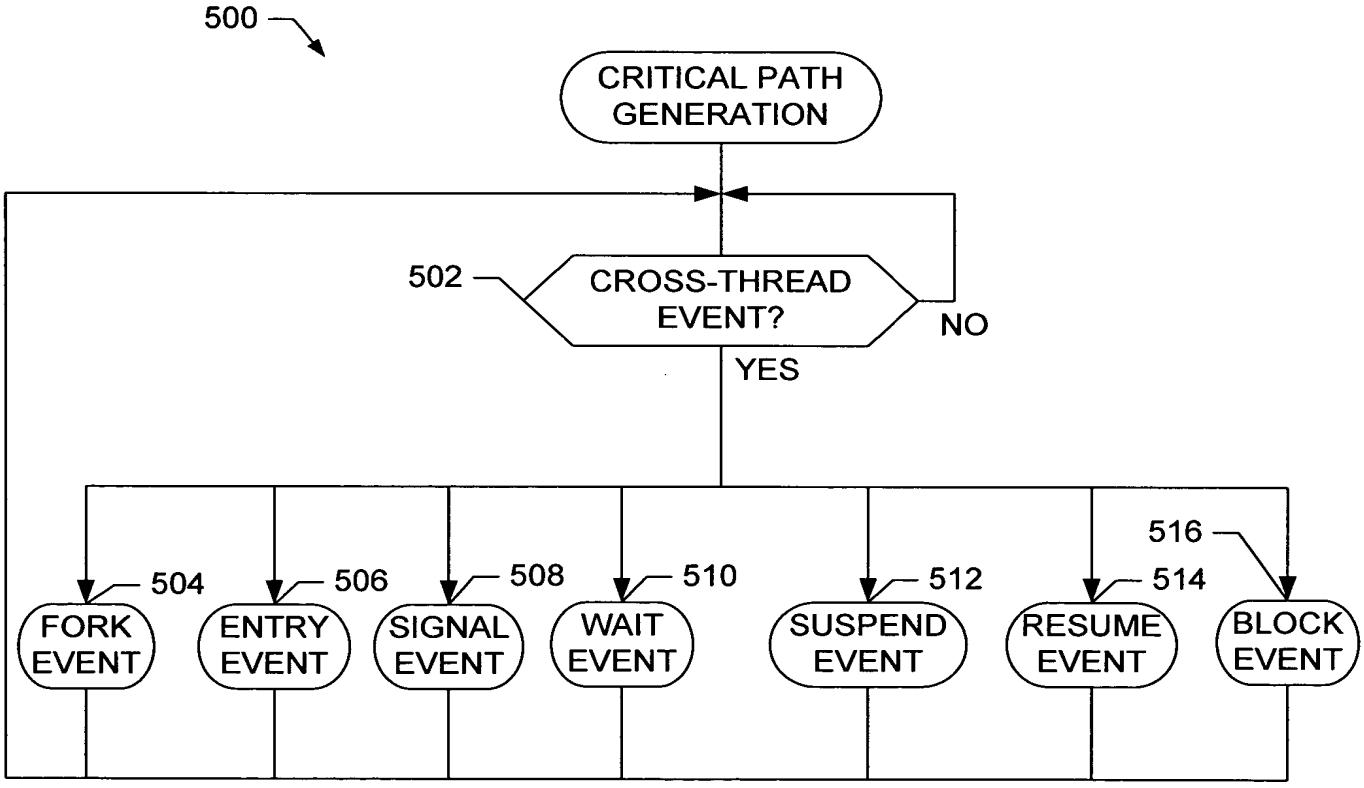
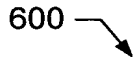

500
CRITICAL PATH GENERATION
502
CROSS-THREAD EVENT?
NO
YES
504
506
508
510
512
514
516
FORK EVENT
ENTRY EVENT
SIGNAL EVENT
WAIT EVENT
SUSPEND EVENT
RESUME EVENT
BLOCK EVENT

FIG. 5



FORK
* CREATE NEW CHILD THREAD OBJECT
* CREATE NEW LEAVES FOR PARENT THREAD AND CHILD THREAD
  - ATTACH CHILD THREAD'S LEAF AS A PENDING LEAF TO THE CHILD THREAD
  - ATTACH PARENT THREAD'S NEW LEAF AS NEW LEAF FOR PARENT THREAD
* EXECUTE CREATE API
* IF CREATE FAILED
  REMOVE CHILD LEAF & DELETE IT

FIG. 6

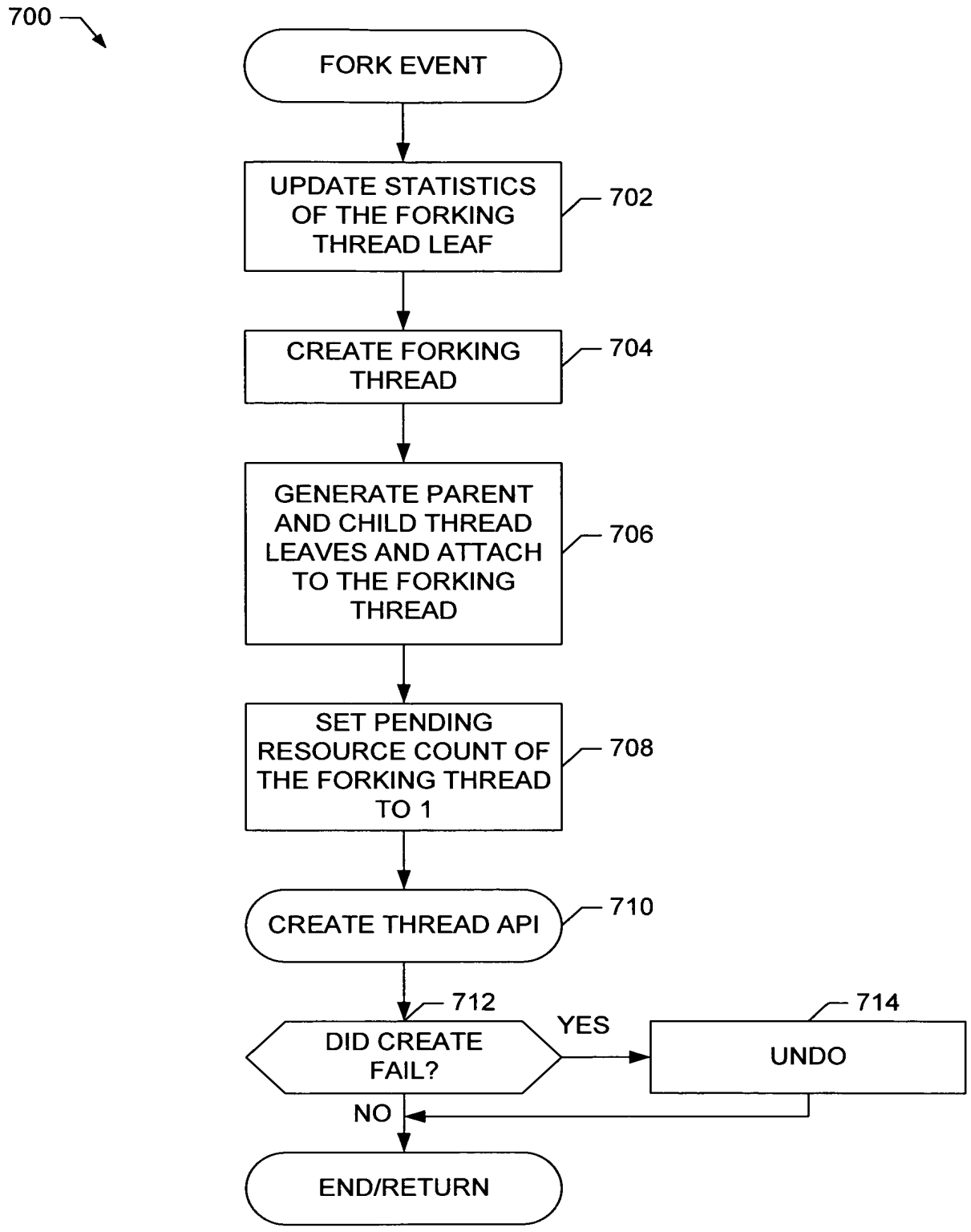

700
FORK EVENT
UPDATE STATISTICS OF THE FORKING THREAD LEAF
702
CREATE FORKING THREAD
704
GENERATE PARENT AND CHILD THREAD LEAVES AND ATTACH TO THE FORKING THREAD
706
SET PENDING RESOURCE COUNT OF THE FORKING THREAD TO 1
708
CREATE THREAD API
710
712
DID CREATE FAIL?
YES
714
UNDO
NO
END/RETURN

FIG. 7

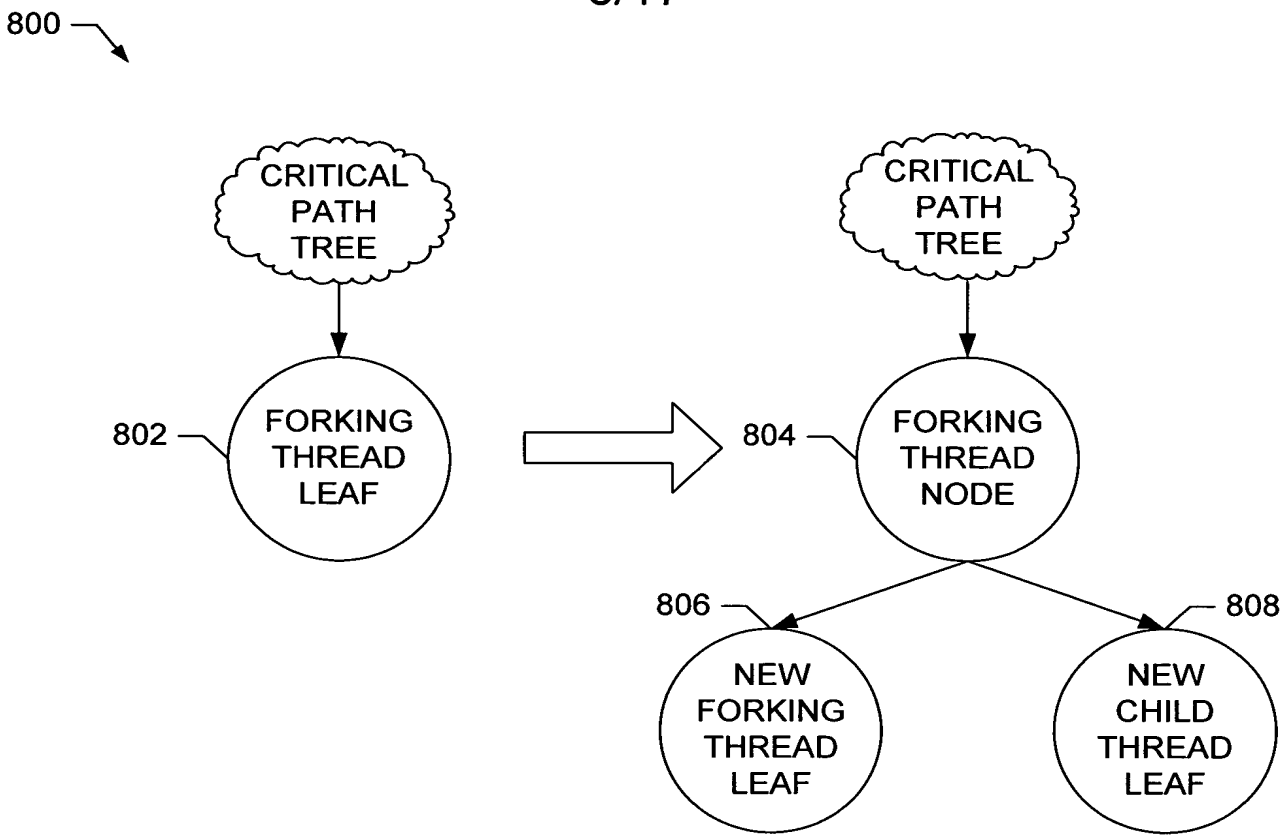


FIG. 8

900

ENTRY
* INCREMENT CONCURRENCY LEVEL
* DID WE MISS THE FORK CALL?
  - CREATE LEAF FOR CHILD BUT DO NOT ATTACH TO ANY PARENT NODE
  - RETURN
* UPDATE STATISTICS OF CHILD THREAD'S LEAF

FIG. 9

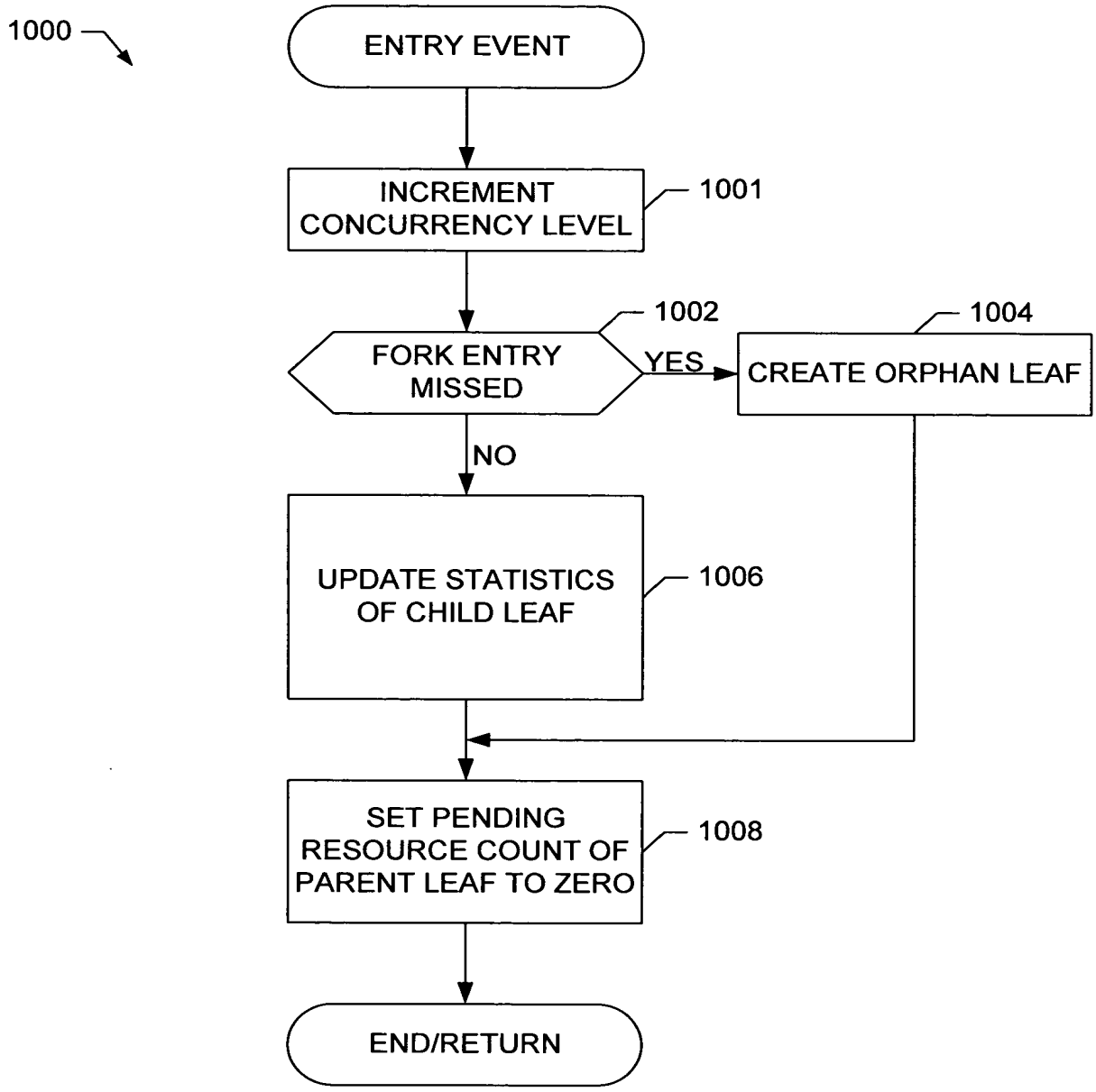

1000
ENTRY EVENT
INCREMENT CONCURRENCY LEVEL
1001
1002
FORK ENTRY MISSED
YES
1004
CREATE ORPHAN LEAF
NO
UPDATE STATISTICS OF CHILD LEAF
1006
SET PENDING RESOURCE COUNT OF PARENT LEAF TO ZERO
1008
END/RETURN

FIG. 10

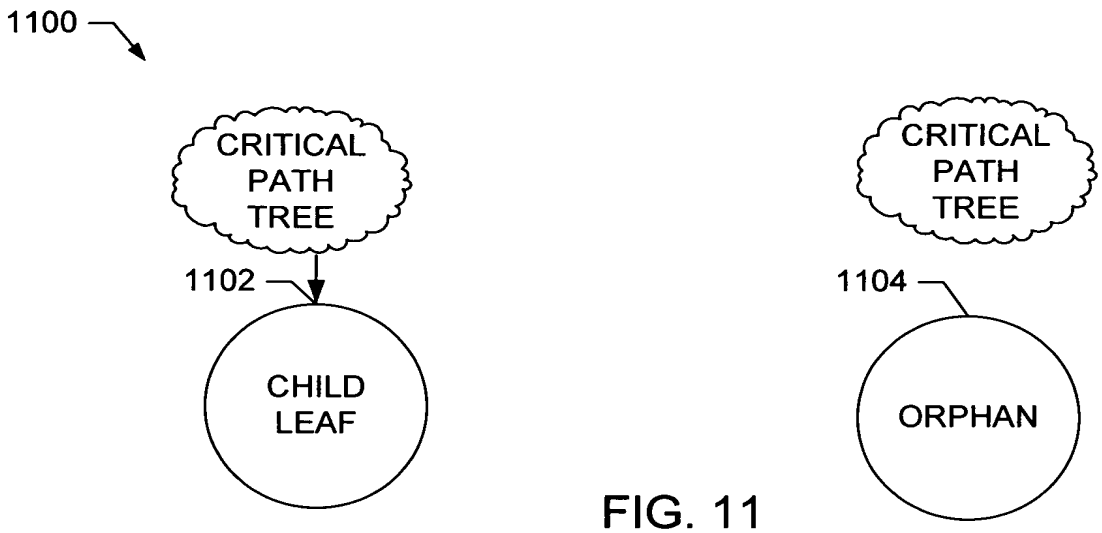


FIG. 11

1200

SIGNAL

```
* GET THE NUMBER OF WAITING THREADS ON SYNC OBJECT
* IF NOT SELF TERMINATING DO THE API CALL
* IF SIGNAL WAS SUCCESSFUL
 - IF THERE WAS AT LEAST ONE WAITING THREAD FOR THIS OBJECT
    - GET THE CURRENT LEAF L OF THE SIGNALING THREAD
    - CREATE NEW LEAF S1 FOR SIGNALING THREAD W/ L AS PARENT NODE
    - CREATE PENDING NODE S2 FOR SIGNALED THREAD W/ L AS PARENT NODE
    - SET RESOURCE CNT OF S2 TO THE COUNT OF RESOURCES BEING SIGNALED
    - SET TIMESTAMP OF S2 TO THE CURRENT TIME
    - IF SIGNALED OBJECT IS A SEMAPHORE
       - IF OBJECT ALREADY HAS A PENDING NODE WITH INFINITE SIGNAL CNT
          - REMOVE S2
       ELSE
          - APPEND S2 TO PENDING NODE LIST OF SEMAPHORE
     ELSE /* NOT A SEMAPHORE */
      - IF THE SIGNALED OBJECT ALREADY HAS ANOTHER PENDING NODE
          - REMOVE S2
       ELSE
          - ADD S2 TO THE PENDING NODE LIST OF THE SIGNALED OBJECT
  ELSE /* NO WAITING THREAD */
    - IF SIGNALED OBJECT IS A SEMAPHORE
       - ADD SIGNAL CNT TO THE TOTAL PENDING RESOURCE COUNT OF
SEMAPHORE SYNC OBJECT

 - IF THIS IS A THREAD TERMINATION OPERATION
    - IF THE TARGET THREAD WAS ACTIVE
       - DECREMENT CONCURRENCY LEVEL
    - SET THREAD STATE TO DEAD
    - IF THERE WAS NO WAITING THREAD
       DELETE LEAF NODE OF TERMINATED THREAD
* IF SELF TERMINATING
    - CALL ACTUAL API NOW
*IF SIGNAL&WAIT, CALL WAIT () ENTRY
```

FIG. 12

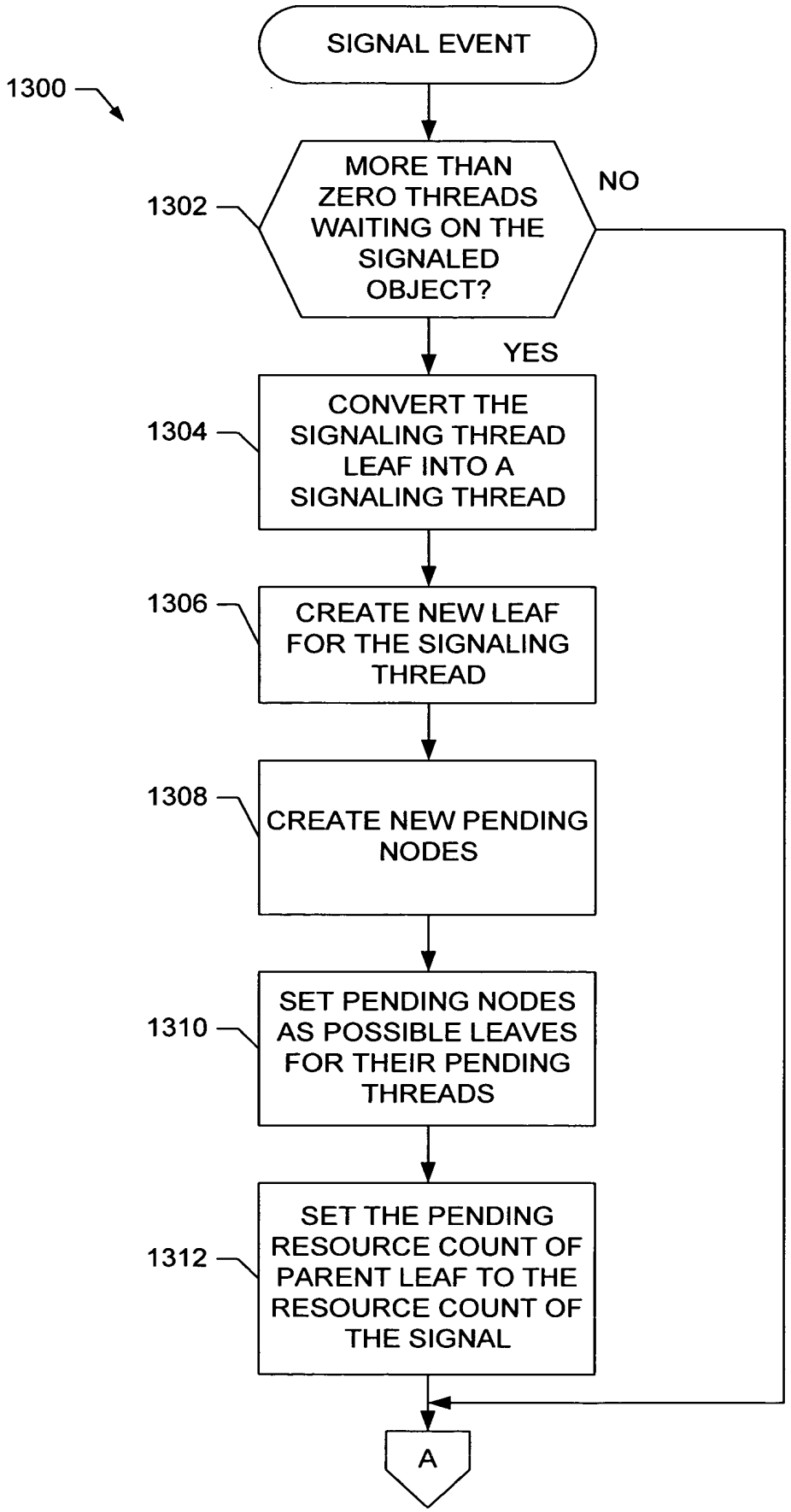

SIGNAL EVENT
1300
MORE THAN ZERO THREADS WAITING ON THE SIGNALED OBJECT?
1302
NO
YES
CONVERT THE SIGNALING THREAD LEAF INTO A SIGNALING THREAD
1304
CREATE NEW LEAF FOR THE SIGNALING THREAD
1306
CREATE NEW PENDING NODES
1308
SET PENDING NODES AS POSSIBLE LEAVES FOR THEIR PENDING THREADS
1310
SET THE PENDING RESOURCE COUNT OF PARENT LEAF TO THE RESOURCE COUNT OF THE SIGNAL
1312
A

FIG. 13A

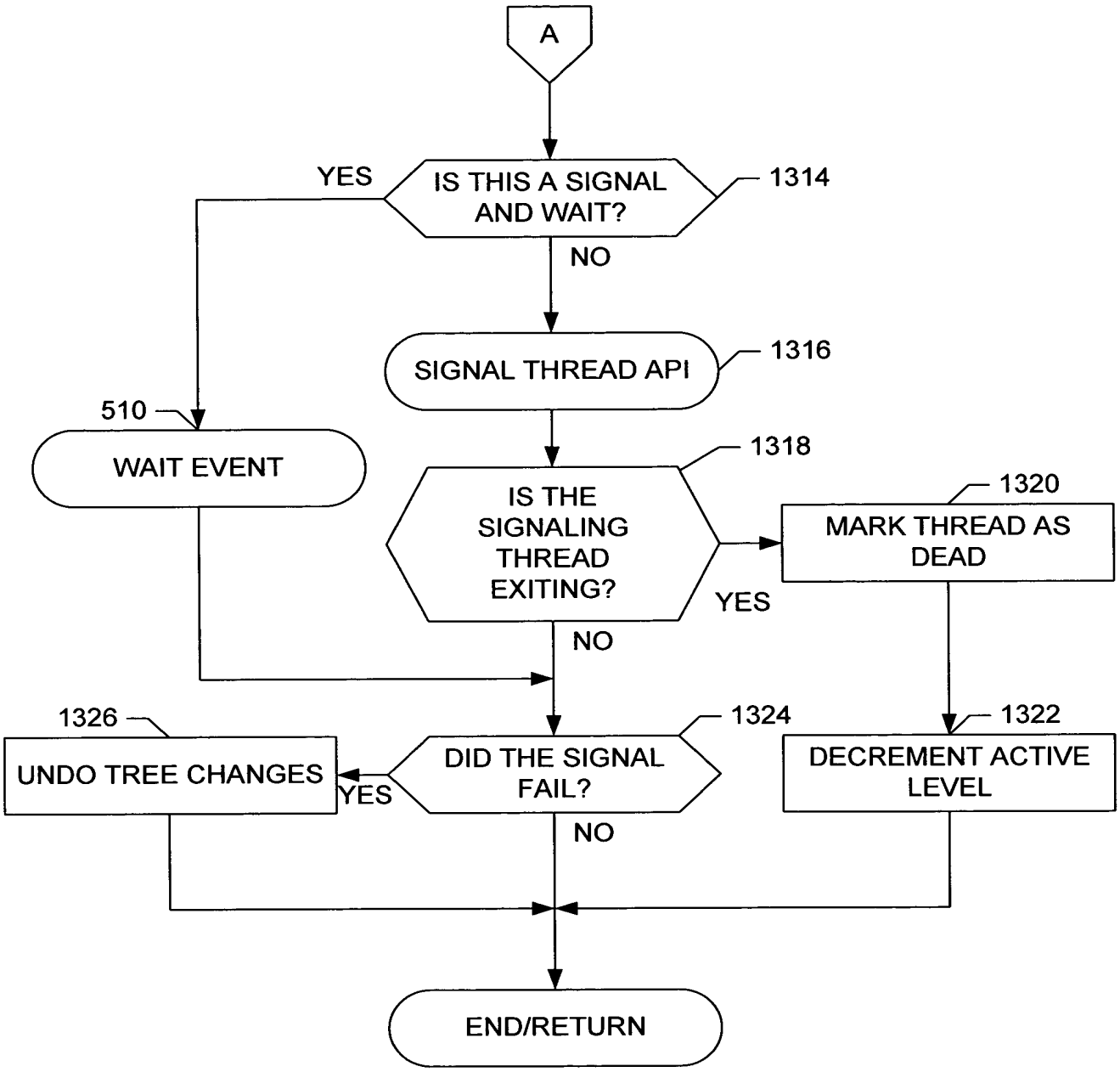

A
IS THIS A SIGNAL AND WAIT?
1314
YES
NO
SIGNAL THREAD API
1316
510
WAIT EVENT
1318
IS THE SIGNALING THREAD EXITING?
1320
MARK THREAD AS DEAD
YES
NO
1326
UNDO TREE CHANGES
YES
DID THE SIGNAL FAIL?
1324
1322
DECREMENT ACTIVE LEVEL
NO
END/RETURN

FIG. 13B

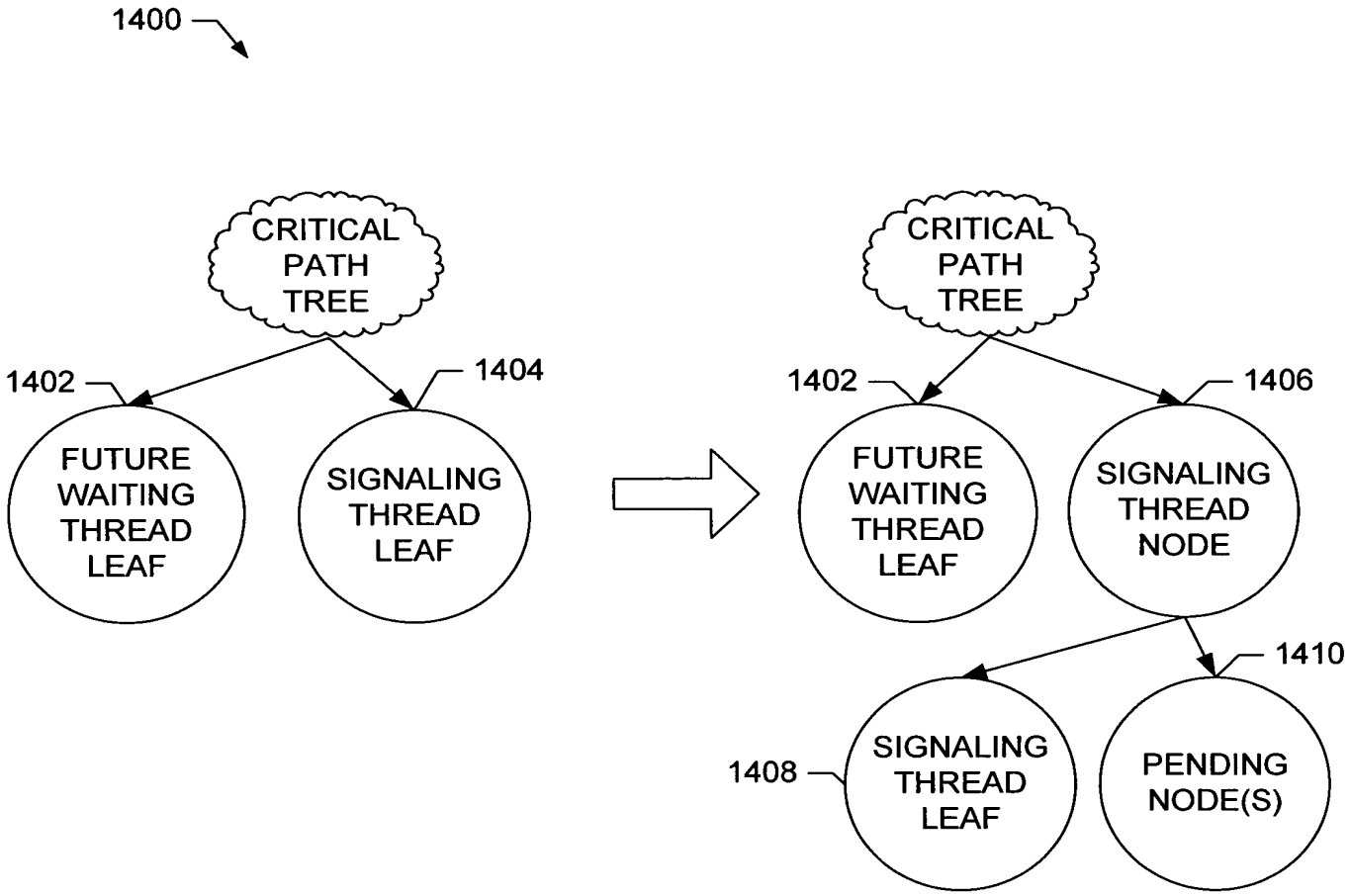

1400
CRITICAL PATH TREE
1402
1404
FUTURE WAITING THREAD LEAF
SIGNALING THREAD LEAF
CRITICAL PATH TREE
1402
1406
FUTURE WAITING THREAD LEAF
SIGNALING THREAD NODE
1410
1408
SIGNALING THREAD LEAF
PENDING NODE(S)

FIG. 14

1500

WAIT

```
* IF BLOCKING WAIT
   - CHANGE WAITING THREAD STATE TO WAIT
    - DECREMENT CONCURRENCY LEVEL
   - FOR EACH OBJECT TO BE WAITED ON
       - REGISTER THAT THIS THREAD IS WAITING FOR THE OBJECT
         (ATOMIC INCREMENT WAIT COUNT OF OBJECT)
* DO THE API CALL
* IF NOT BLOCKING WAIT
   - DONE
 * INCREMENT CONCURRENCY LEVEL
* FOR EACH OBJECT THIS THREAD WAITED ON
   - REGISTER THAT THIS THREAD NO LONGER WAIT FOR THE OBJECT
     (ATOMIC DECREMENT WAIT COUNT OF OBJECT)
* IF WAIT FAILED OR TIMED OUT
   - UPDATE LEAF OF CURRENT (WAITING) THREAD WITH TIME SPENT WAITING AS
BLOCKING TIME
  ELSE /* WAIT DIDN'T FAIL */
   - FOR EACH OBJECT THIS THREAD WAITED ON
       /* CLAIM A LEAF FROM EACH OF OBJECT */
       - GET A PENDING NODE FROM THE OBJECT
       - IF THE RESOURCE COUNT OF NODE IS NOT INFINITE
          - DECREMENT RESOURCE COUNT
       - IF COUNT > 0
          - DUPLICATE THE PENDING NODE AND ADD TO A LIST OF POTENTIAL LEAVES
   - SELECT A POTENTIAL LEAF WITH A LATEST TIMESTAMP AND REMOVE THE REST
   - IF THE WAITING THREAD HAS A VALID RESUME LEAF (CREATED VIA A RESUME
ENTRY POINT) WHOSE TIMESTAMP IS LATER THAN THE CURRENT POTENTIAL LEAF
       - MAKE IT THE NEW POTENTIAL LEAF AND REMOVE THE OLD POTENTIAL LEAF
     ELSE
       - REMOVE THE RESUME NODE
   - IF THE WAITING THREAD'S PREVIOUS LEAF'S TIMESTAMP IS LATER THAN THE
CURRENT POTENTIAL LEAF
       - MAKE IT THE NEW POTENTIAL LEAF AND REMOVE THE OLD POTENTIAL LEAF
     ELSE
       - REMOVE THE OLD NODE
   - IF THERE IS A POTENTIAL LEAF
       - MAKE IT THE NEW LEAF FOR THE THREAD
   - IF THE THREAD'S NEW LEAF IS NOT THE THREAD'S OLD LEAF
       UPDATE STATS OF THE NEW LEAF
* SET THREAD TO ACTIVE
```

FIG. 15

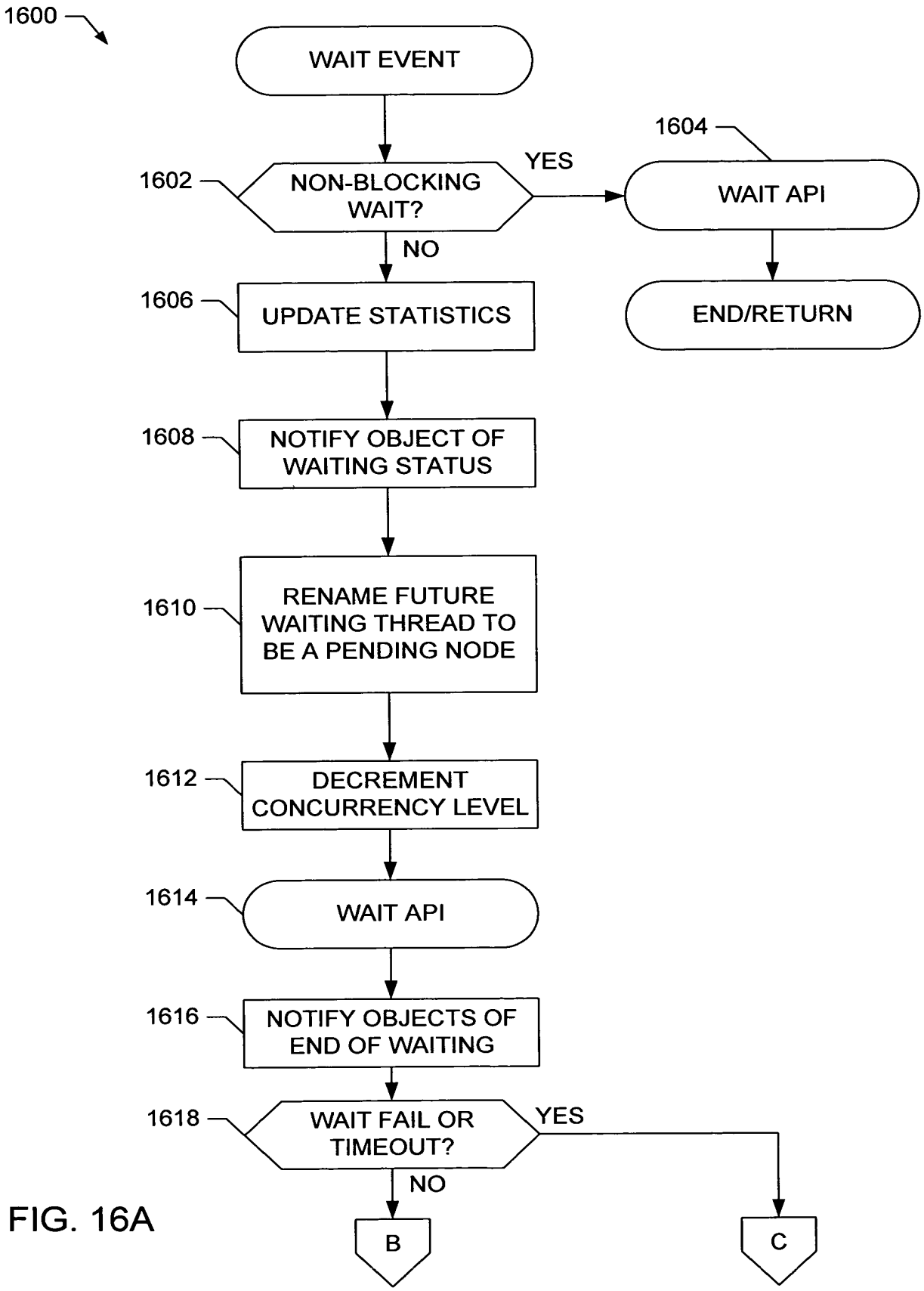

1600
WAIT EVENT
1602
NON-BLOCKING WAIT?
YES
1604
WAIT API
NO
END/RETURN
1606
UPDATE STATISTICS
1608
NOTIFY OBJECT OF WAITING STATUS
1610
RENAME FUTURE WAITING THREAD TO BE A PENDING NODE
1612
DECREMENT CONCURRENCY LEVEL
1614
WAIT API
1616
NOTIFY OBJECTS OF END OF WAITING
1618
WAIT FAIL OR TIMEOUT?
YES
NO
B
C

FIG. 16A

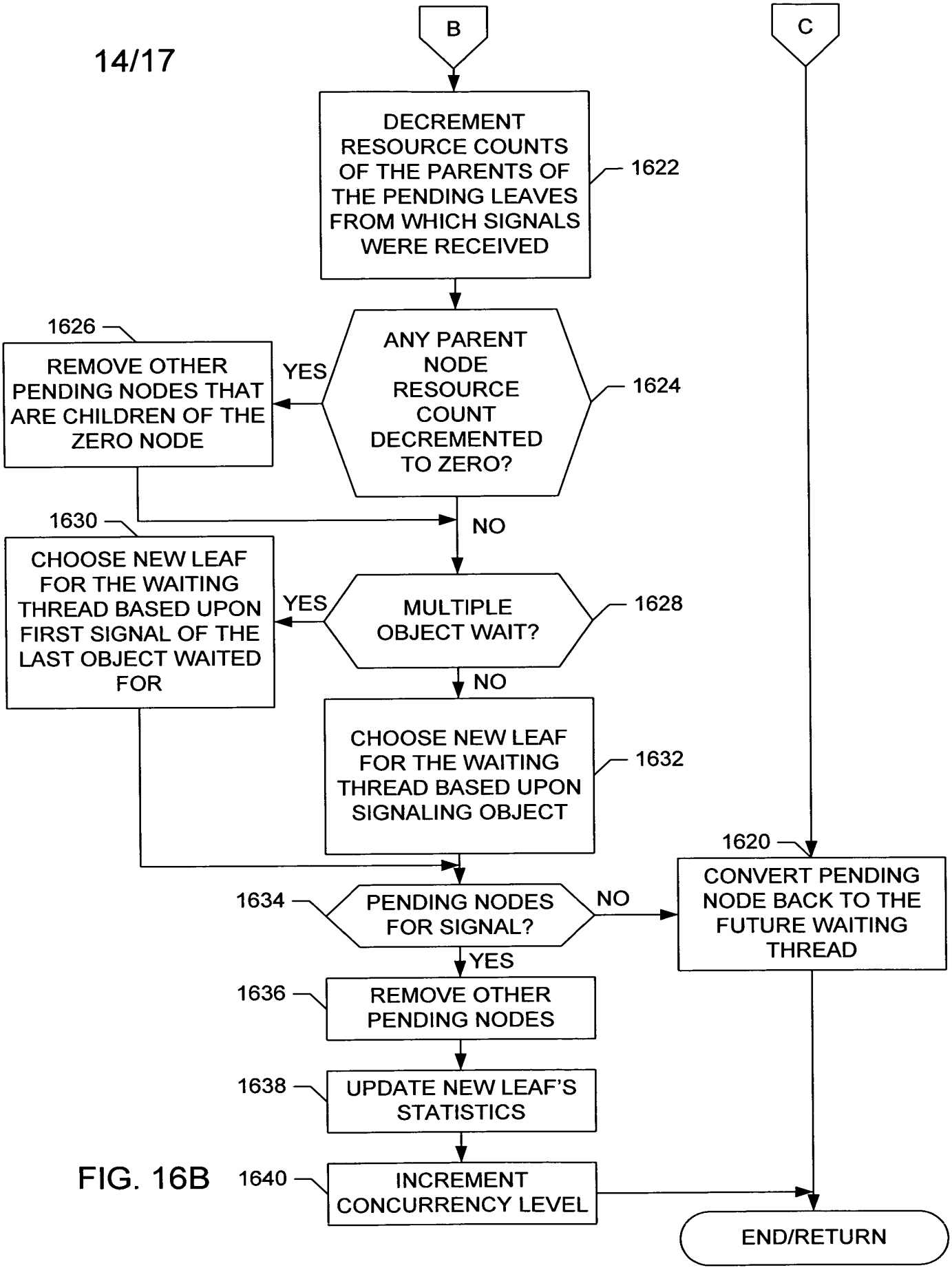

B
C
DECREMENT RESOURCE COUNTS OF THE PARENTS OF THE PENDING LEAVES FROM WHICH SIGNALS WERE RECEIVED
1622
ANY PARENT NODE RESOURCE COUNT DECREMENTED TO ZERO?
1624
YES
1626
REMOVE OTHER PENDING NODES THAT ARE CHILDREN OF THE ZERO NODE
NO
MULTIPLE OBJECT WAIT?
1628
YES
1630
CHOOSE NEW LEAF FOR THE WAITING THREAD BASED UPON FIRST SIGNAL OF THE LAST OBJECT WAITED FOR
NO
CHOOSE NEW LEAF FOR THE WAITING THREAD BASED UPON SIGNALING OBJECT
1632
PENDING NODES FOR SIGNAL?
1634
NO
1620
CONVERT PENDING NODE BACK TO THE FUTURE WAITING THREAD
YES
REMOVE OTHER PENDING NODES
1636
UPDATE NEW LEAF'S STATISTICS
1638
INCREMENT CONCURRENCY LEVEL
1640
END/RETURN

FIG. 16B

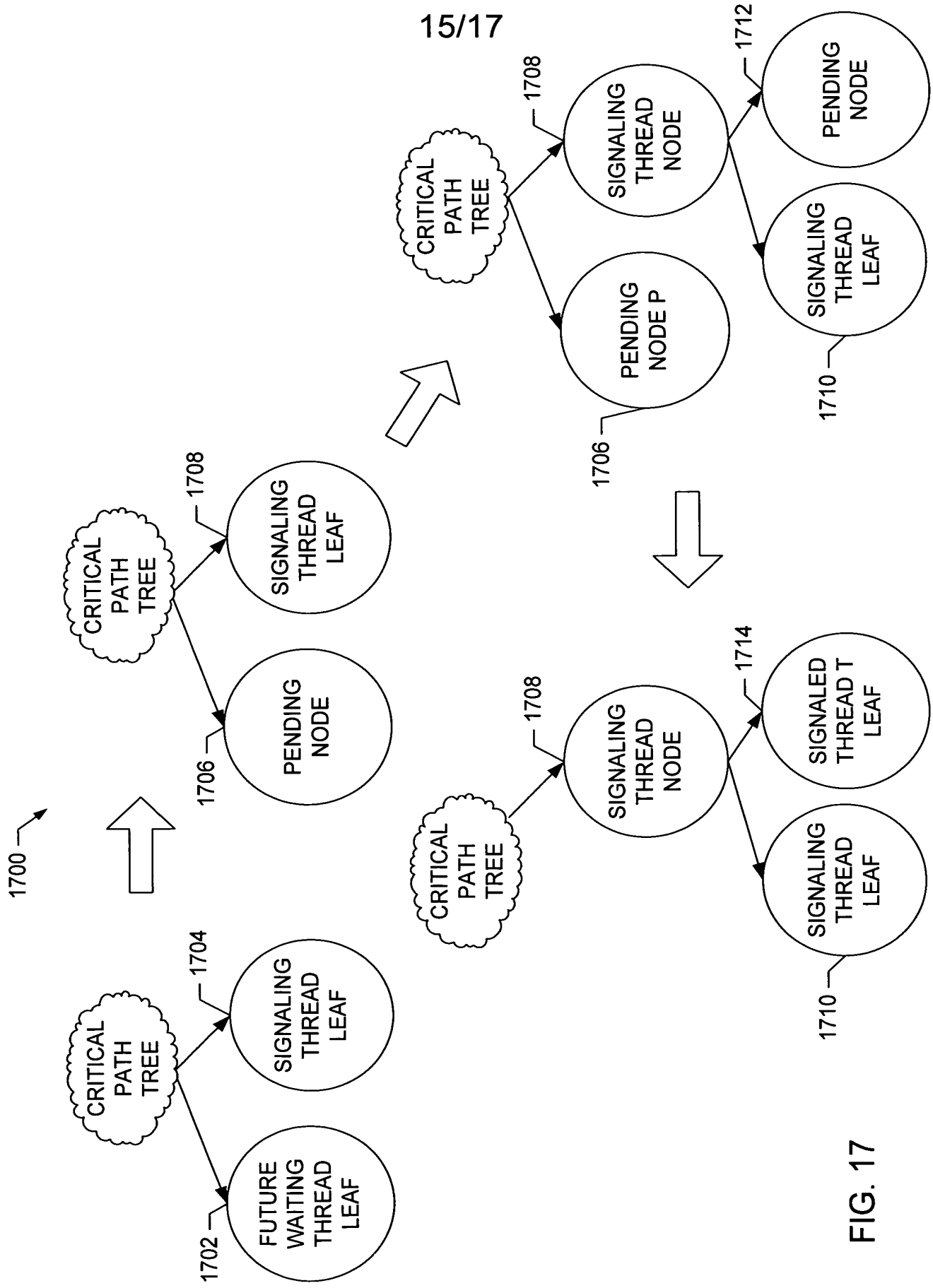

1700
CRITICAL PATH TREE
FUTURE WAITING THREAD LEAF
1702
SIGNALING THREAD LEAF
1704
PENDING NODE
1706
SIGNALING THREAD LEAF
1708
CRITICAL PATH TREE
PENDING NODE P
1706
SIGNALING THREAD NODE
1708
PENDING NODE
1712
SIGNALING THREAD LEAF
1710
CRITICAL PATH TREE
SIGNALING THREAD NODE
1708
SIGNALED THREAD T LEAF
1714
SIGNALING THREAD LEAF
1710

FIG. 17

1800

SUSPEND
* IF THE TARGET THREAD IS NOT ALREADY SUSPENDED
  - SET THE TIMESTAMP THAT THE THREAD IS SUSPENDED
* ACTUALLY DO THE API

FIG. 18

1900

RESUME
* GET TIMESTAMP T BEFORE SIGNALING
* GET THE TIME THAT THE TARGET THREAD WAS SUSPENDED
* ACTUALLY DO THE API
* IF THE TARGET THREAD WAS NOT SUSPENDED
  - CLEAR SUSPENDED TIME OF TARGET THREAD
  ELSE IF THE TARGET THREAD WAS SUSPENDED BUT NOW RESTARTED
  - GET LEAF L OF THE RESUMING THREAD
  - CREATE A NEW LEAF FOR THE RESUMING THREAD W/ L AS PARENT NODE
  - CREATE A RESUME NODE FOR TARGET THREAD (WITH TIMESTAMP T) W/ L AS PARENT NODE
  - REPLACE ANY OLD UNCLAIMED RESUME NODE OF TARGET THREAD WITH NEW NODE
  - IF AN OLD UNCLAIMED RESUME NODE EXISTS
    - REMOVE IT
  - IF THE TARGET THREAD WAS ACTIVE
    - USE THE TARGET THREAD'S NEW RESUME NODE AS ITS NEW LEAF & REMOVE OLD LEAF
    - UPDATE STATS OF THREAD'S NEW LEAF
    - SET TARGET THREAD STATE TO ACTIVE
    - INC CONCURRENCY LEVEL

FIG. 19

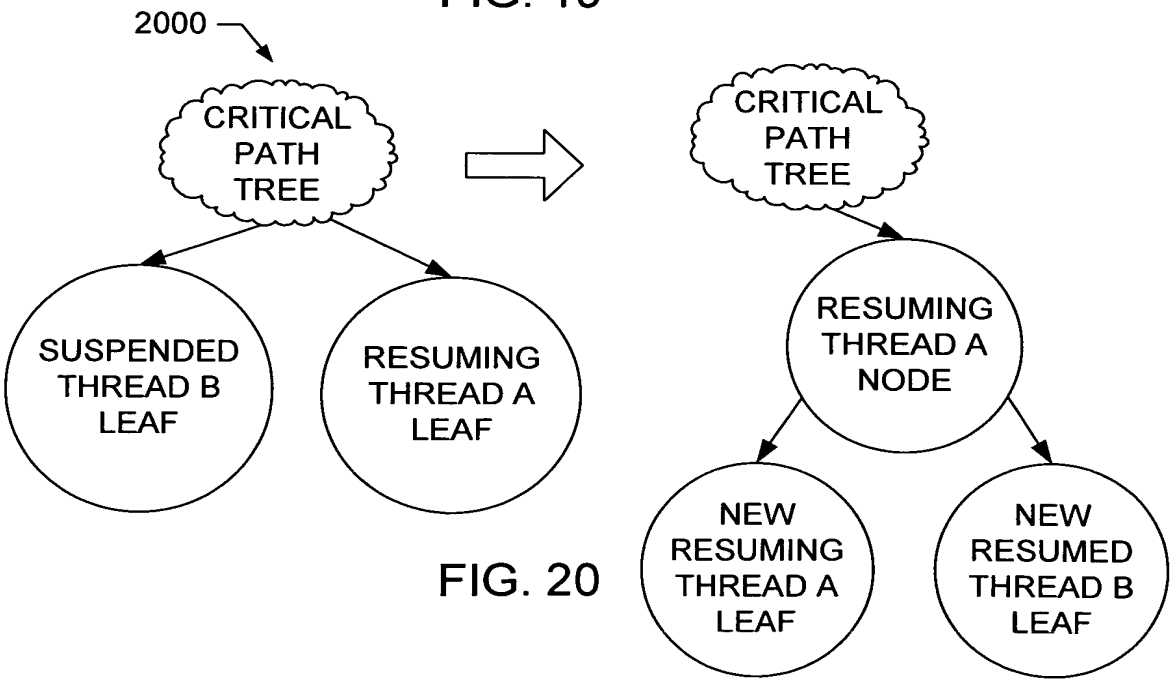


FIG. 20

2100

BLOCK
* SET CURRENT THREAD STATE TO BLOCK
 * DECREMENT CONCURRENCY LEVEL
* DO API
 * INCREMENT CONCURRENCY LEVEL
 * IF THREAD NOW HAS A VALID RESUME LEAF (CREATED IN RESUME ENTRY POINT)
  - REMOVE CURRENT THREAD'S OLD LEAF
  - USE RESUME LEAF AS THE THREAD'S NEW LEAF
* UPDATE STATS OF THREAD'S LEAF
 * SET THREAD STATE TO ACTIVE

FIG. 21

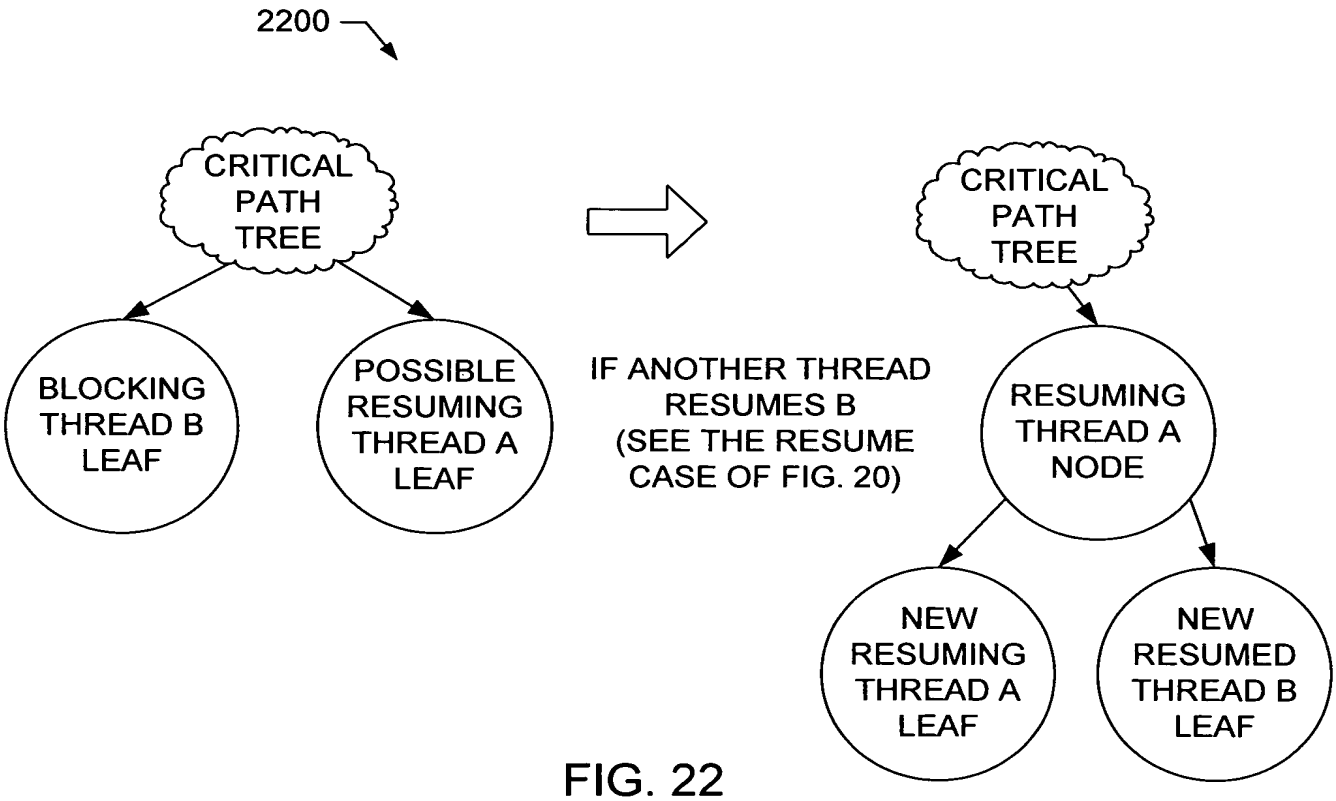


FIG. 22